Research on Flu using Twitter Data

Fei Zhao, Liangji Wang, Xinyi Chen

1. Introduction

There are more than 3 million cases of severe illness and about 500,000 deaths are caused by seasonal influenza epidemics worldwide each year [1]. Research on seasonal influenza epidemics such as H1N1 is very crucial to reducing its harm and impact, also has significant importance for public health. Studies have shown that if an early detection is made, preventive measures can be taken to contain epidemics [2], [3]. Therefore, it is important to analyze seasonal influenza epidemics data.

2. Related Work

There have been efforts in tracking and predicting flu trends using internet data such as google search patterns. For example, Google Flu Trends estimates Flu and Dengue fever based on search patterns data mining, which lacks of machine learning analysis. We think examining social media trends through Twitter would yield interesting results because it is closely associated with people's lives.

There are lots of studies conducted on different social networks such as Facebook, Flickr, Linkedin, Wikipedia and Youtube. Social media, Twitter in particular, is interesting because people post a lot of live data about things around them. For example, Twitter's chatter feature, a simple model built from the rate at which tweets are created about particular topics, has been used to outperform market-based predictors on box office revenues [4]. Also Twitter data has been used as sensors to build an autonomous earthquake reporting system in Japan [5]. Therefore, analyzing twitter data associated with flu can be interesting.

3. Data Collection

We use Twitter Streaming API to get the data from Twitter. First we perform the setup, which involves creating Twitter app to receive the API key/access token necessary to scrape data from Twitter. Then we connect to Twitter Streaming API and download the data, by using the Tweepy library to connect to the Twitter Streaming API and extract data associated with keywords “flu, cough, fever, cold”

Data set we gtt at last: 1.25GB, 252,291 tweets; we do some visualizations to analyze the data we get, here are the results:

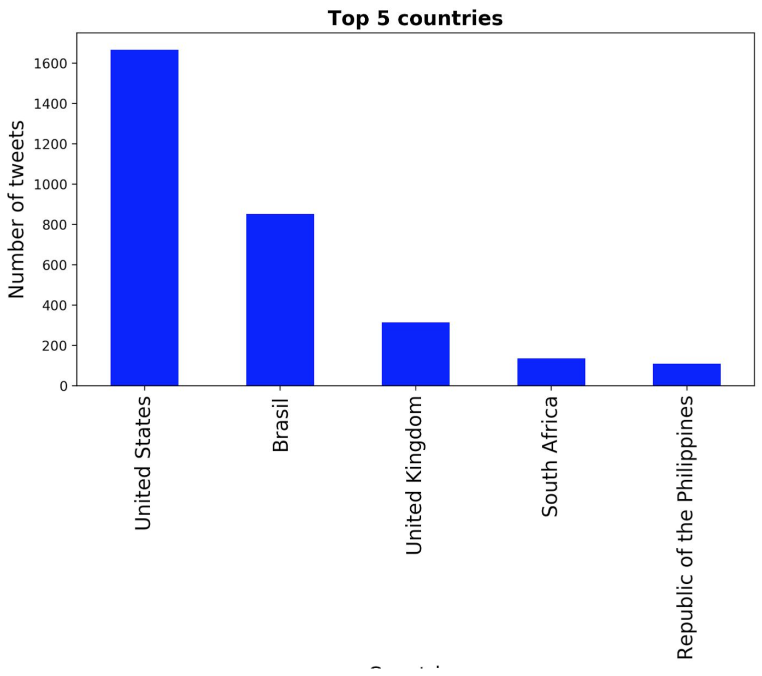


fig.3.1 Top 5 countries

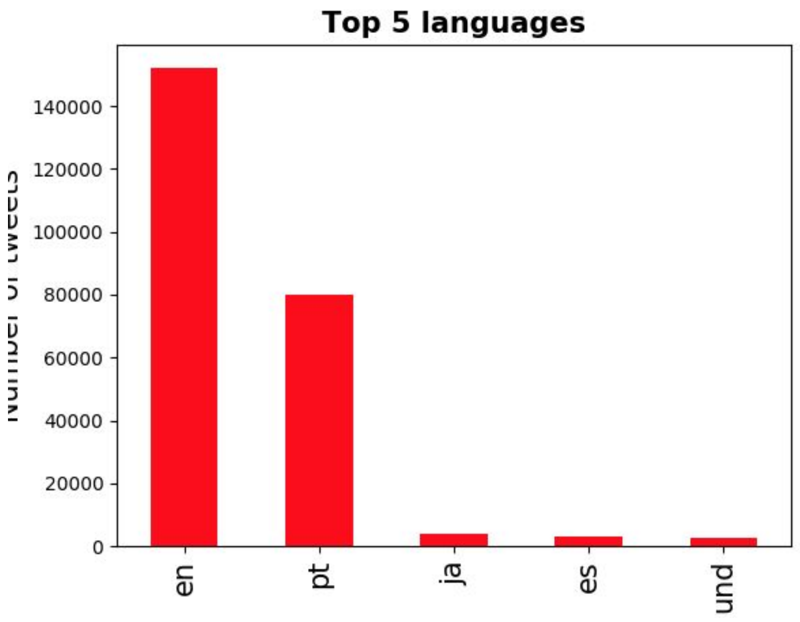


fig.3.2 Top 5 languages

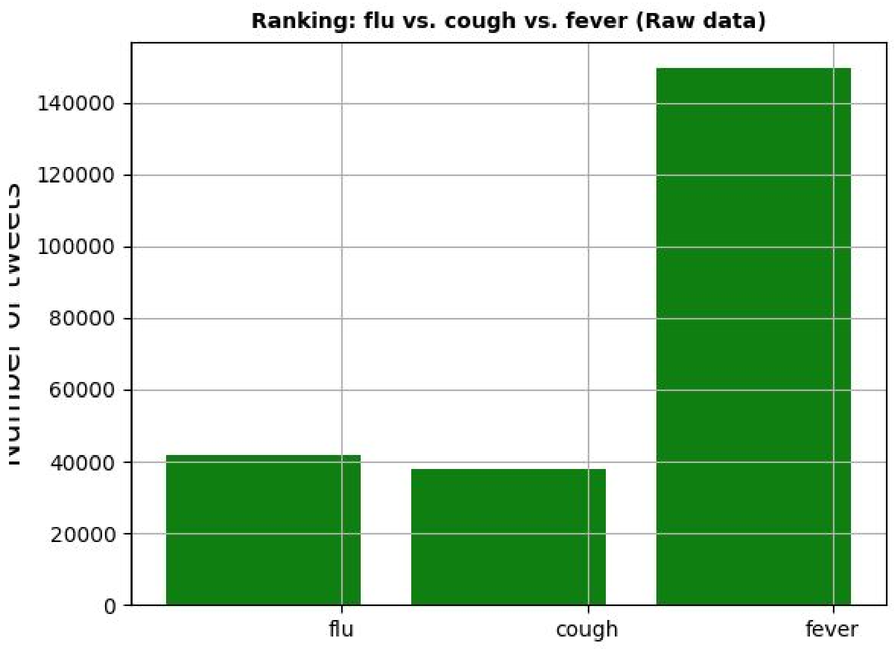


fig.3.3 Flu vs. Cough vs. Fever

4. Experiments Setup

4.1. Spam Elimination

There are lots of spam in our collected data set; for example, some tweets include links to advertising or malicious websites, some are retweeted tweets, which means we collect the same tweets repeatedly. To deal with this spam problem, we do the spam elimination first before experiments.

First, we load the raw data tweet by tweet, then check if this tweet includes links or is a retweeted tweet, if so we discard this tweet; otherwise, we keep this tweet to our clean data set.

After the spam elimination process, there are 206,243 clean tweets left for our experiments.

Core Code:

```
tweets_file = open(readfile, "r")
  outfile = open(outfile, 'a')
  for line in tweets_file:
      try:
            tweet = json.loads(line)
            if 'http' not in tweet['text'] and
tweet['retweeted'] == False:
                json.dump(tweet, outfile)
                outfile.write('\n')
        except:
            Continue
```

4.2. Build Dictionary

Build a dictionary with american state and extract those tweets based on this dictionary, and then count the amount for each state and plot the distribution.

Core Code:

```
for line in tweets_file:
        try:
            temp = json.loads(line)
            tweet = {}
            tweet['text'] = temp['text']
            tweet['location'] = temp['location']
            tweet['timestamp_ms'] =
temp['timestamp_ms']
            tweet['lang'] = temp['lang']
            tweets_data.append(tweet)
        except:
            continue
```

5. Experiments

5.1. Analysis Based on Time Periods

Based on the geolocation dictionary, we can generate a time dictionary with tweets in America, and count the amount based on each day and hour, then plot the distribution.

Core Code:

```
def timeExp(dataPath = 'cleanData.json'):
    dateDict, hourDict = getTimeDict(dataPath)
    plotTimeDict(dateDict)
    for key in hourDict:
        plotTimeDict(hourDict[key])
```

Results:

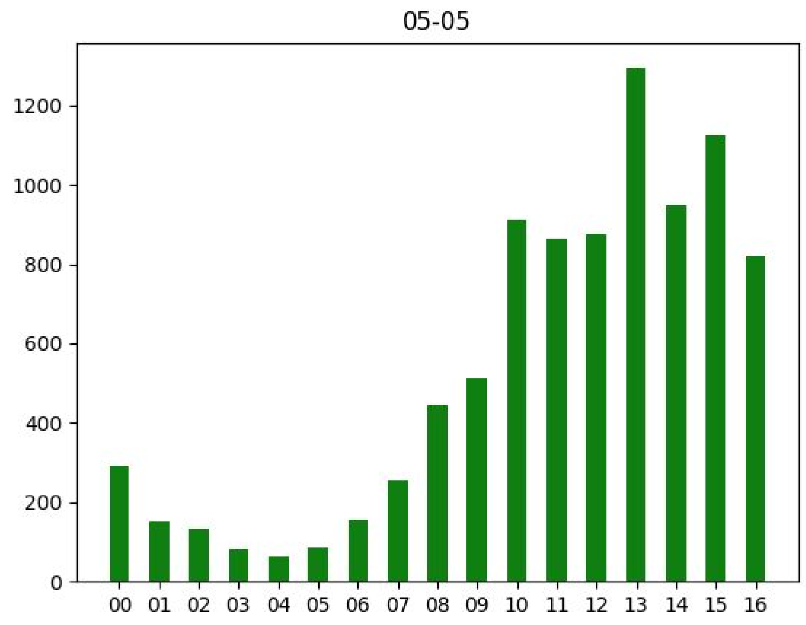


fig.5.1.1 Number of tweets at different times of day

5.2. Analysis Based on Locations

Build a dictionary with american state and extract those tweets based on this dictionary, then count the amount for each state and plot the distribution.

Core Code:

```
def geoExp(dataPath = 'cleanData.json'):
    tweets = readData(dataPath)
    geoDict, geoTweets = getGeoDict(tweets)
    vmin = 0
```

```
vmax = 0
for key in geoDict:
    print key, geoDict[key]
    if geoDict[key] > vmax:
        vmax = geoDict[key]
plotGeoDict(geoDict)
plotGeo(geoDict)
```

Results:

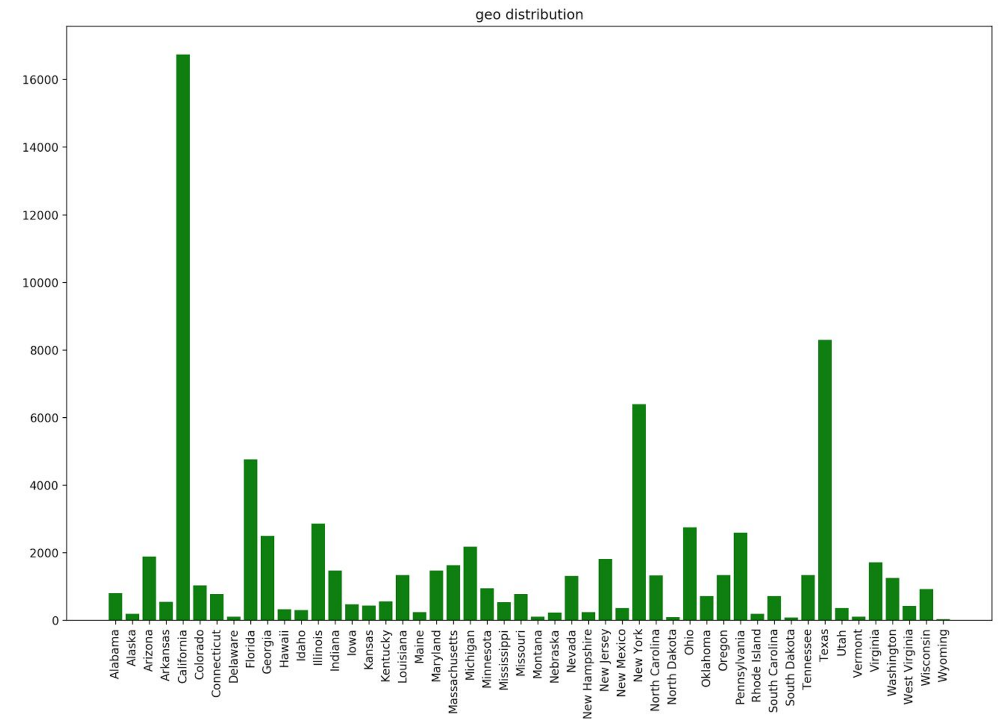


fig.5.2.1 The geo distribution of tweets (state)

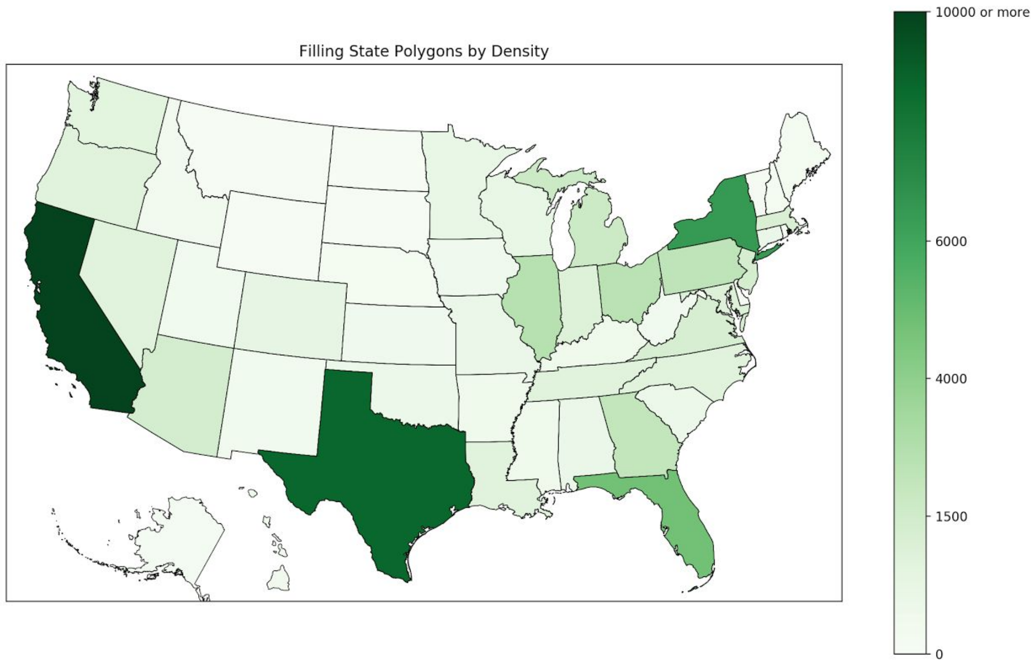


fig.5.2.2 Map view of geo distribution experiments result

As we can see the number of flu tweets and population has a positive correlation; most tweets are from California, this is basically because CA has the largest population.

5.3. Sentiment Analysis

By looking into the text content of some tweets, it could be obviously observed that lots of them has a sentiment. We decided to do a sentiment analysis to dig into those sentences. For achieving the goal, we used the TextBlob which is a popular NLP api. Text Blob has many useful functions and one of them is sentiment analysis.

The TextBlob api is very straightforward to use. The sentiment property returns a namedtuple of the form **`Sentiment(polarity, subjectivity)`**. The polarity score is a float within the range [-1.0, 1.0]. The subjectivity is a float within the range [0.0, 1.0] where 0.0 is very objective and 1.0 is very subjective. The format of the data was transferred to csv for a higher readability.

The mainly input here is the text contents from the twitter api. After tried some simple examples, we determined the thresholds of the sentiment analysis: for the polarity, the contents between (-0.1,0.1) went to neutral, the ones beyond 0.1 went to positive and vise versa. For the subjectivity, ones beyond 0.5 went to subjective and vise versa.

Core Code:

```
from textblob import TextBlob

for row in data:
   testimonial = TextBlob(row[1].decode('utf-8'))
   if testimonial.sentiment.polarity < -0.1:
        row.append("Negative")
   elif testimonial.sentiment.polarity > 0.1:
        row.append("Positive")
   else:
        row.append("Neutral")
   if testimonial.sentiment.subjectivity > 0.5:
        row.append("Subjective")
   elif testimonial.sentiment.subjectivity < 0.5:
        row.append("Objective")
   else:
        row.append("Neutral")
   csv_sentiment.writerow(row)
```

We decoded the timestamp as we mentioned in previous section. We now mapping the timestamps to four time chunks, morning (6,12), afternoon (12,18), evening (18, 24) and night (0, 6). We only focused on the time-sentiment analysis.

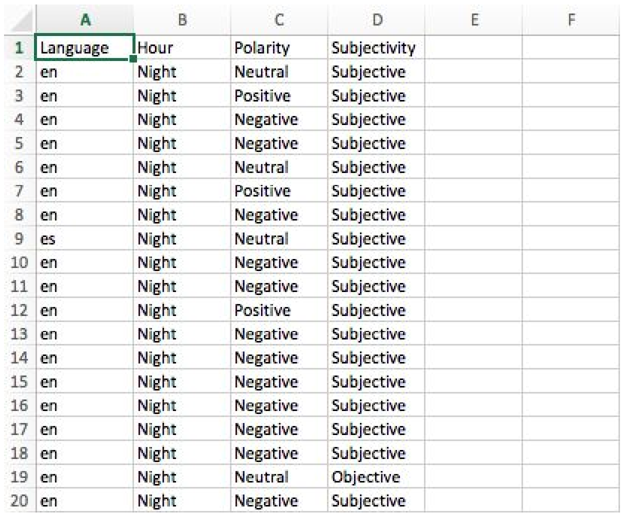
Core Code:

```
import datetime
for row in data:
   time =
datetime.datetime.fromtimestamp(long(row[1])/1000).
strftime('%H')
   time = int(time)
   if 6 <= time < 11:
        row[1] = ("Morning")
        data_with_time.writerow(row)
   elif 12 <= time < 18:
        row[1] = ("Afternoon")
```

```
            data_with_time.writerow(row)
    elif 18 <= time < 24:
            row[1] = ("Evening")
            data_with_time.writerow(row)
    else:
            row[1] = ("Night")
            data_with_time.writerow(row)
```

Finally we got rid of the text itself to have a better performance.

The data file looks like this after the pre-work. We will describe the result in next section.



	A	B	C	D	E	F
1	Language	Hour	Polarity	Subjectivity		
2	en	Night	Neutral	Subjective		
3	en	Night	Positive	Subjective		
4	en	Night	Negative	Subjective		
5	en	Night	Negative	Subjective		
6	en	Night	Neutral	Subjective		
7	en	Night	Positive	Subjective		
8	en	Night	Negative	Subjective		
9	es	Night	Neutral	Subjective		
10	en	Night	Negative	Subjective		
11	en	Night	Negative	Subjective		
12	en	Night	Positive	Subjective		
13	en	Night	Negative	Subjective		
14	en	Night	Negative	Subjective		
15	en	Night	Negative	Subjective		
16	en	Night	Negative	Subjective		
17	en	Night	Negative	Subjective		
18	en	Night	Negative	Subjective		
19	en	Night	Neutral	Objective		
20	en	Night	Negative	Subjective		

fig.5.3.1 The final csv file

We can get the sentiment analysis results; as we can see in the fig.5.3.2, most of tweets are neutral (49.9%), 27.9% of them are negative, and 22.1% of them are positive; also most of them are objective (60.5%).

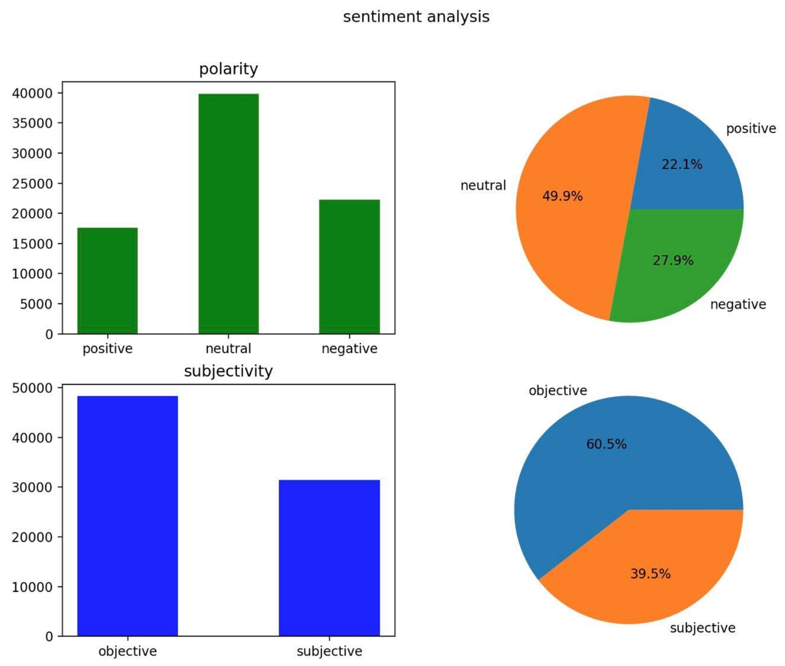


fig.5.3.2 Sentiment analysis results

5.4. Insights and Machine Learning Exploration

After getting the data, we found that it may be interesting and meaningful to apply some machine learning techniques to do a exploration. Hence, we did some data preprocessing work in the last section.

We used the weka to do the work. Before the machine learning. Let take a look at the data itself.

In fig.5.4.1, we could find that among the four chunks of time (left to right: morning, afternoon, evening, night), afternoon is the most active one. In fig.5.4.2, we could find that the among all four time chunks, the numbers of subjective tweets (blue) are roughly the same. However the objective tweets of afternoon is significantly larger than other three. In fig.5.4.3, we could find that among all the collected tweets, the neutral one (blue) has the largest number, compared to the positive (red) and the negative (green). In fig.5.4.4, we could find that for the neutral tweets (very left), the objective ones (red) dominated. For positive tweets (middle), it is nearly half-half. For negative tweets (right), subjective (blue) dominated. This one is really make sense to us since if you are in a bad mood, it is easy to have a subjective claim for something.

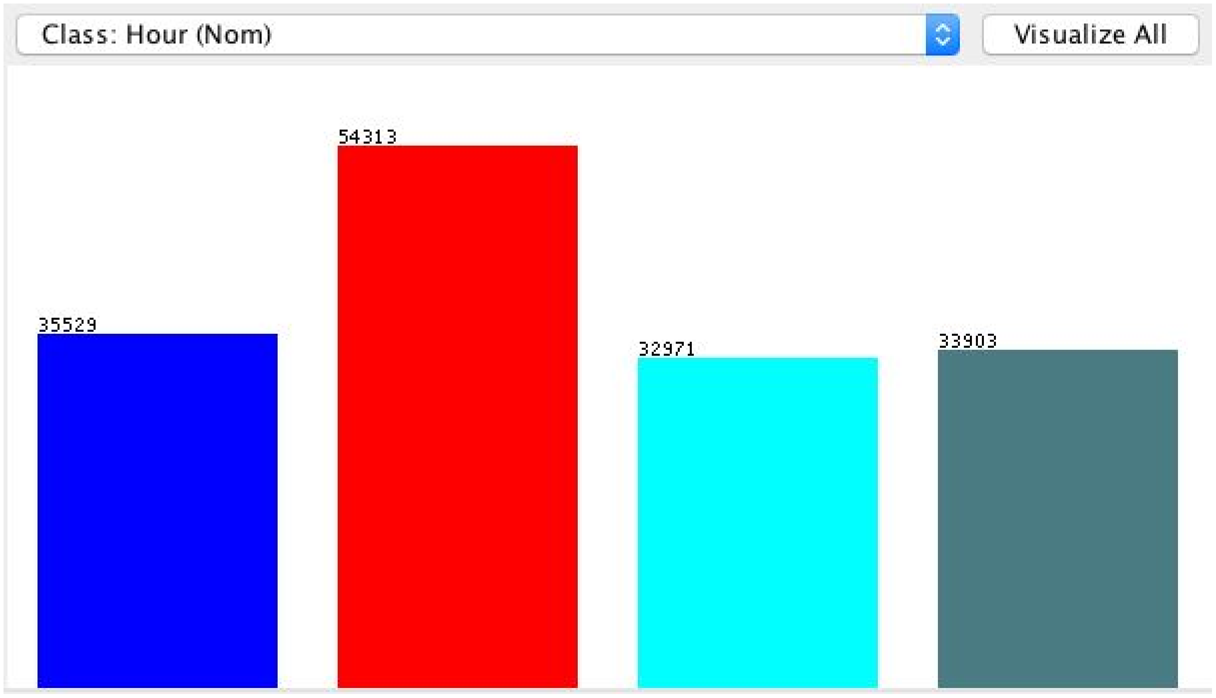


Fig.5.4.1 Number of tweets at different time chunks of day

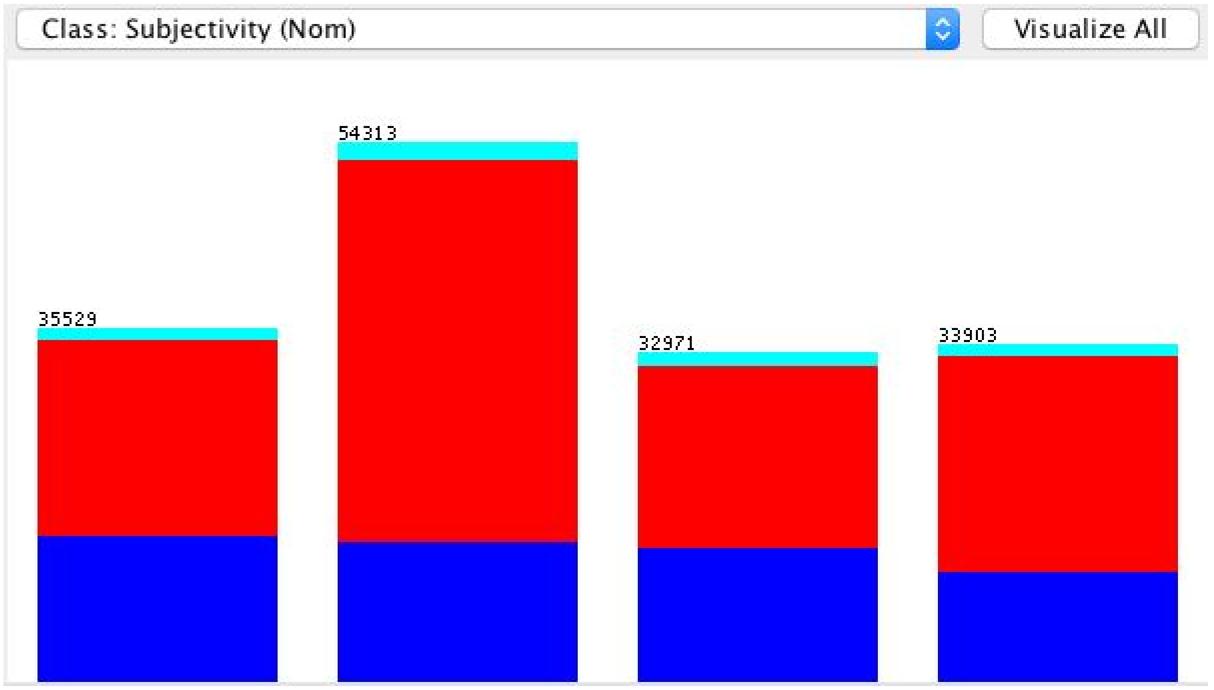


Fig.5.4.2 Number of different subjectivity tweets at different times of day

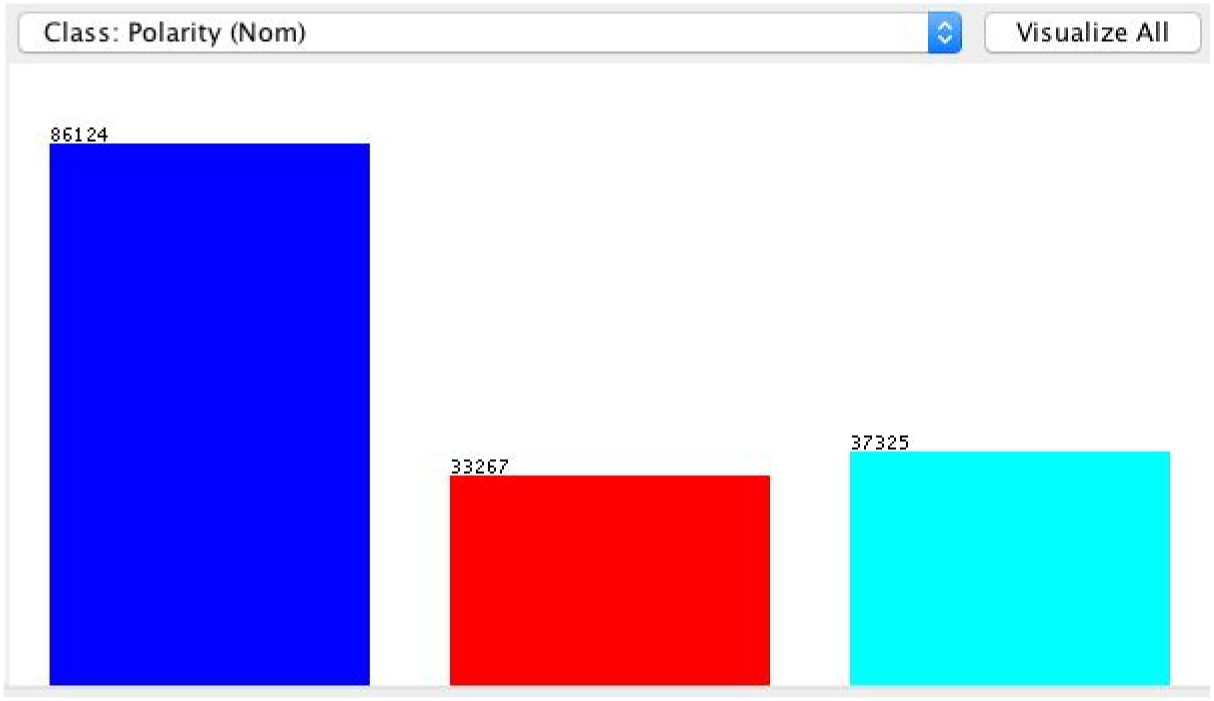


Fig.5.4.3 Number of different polarity tweets

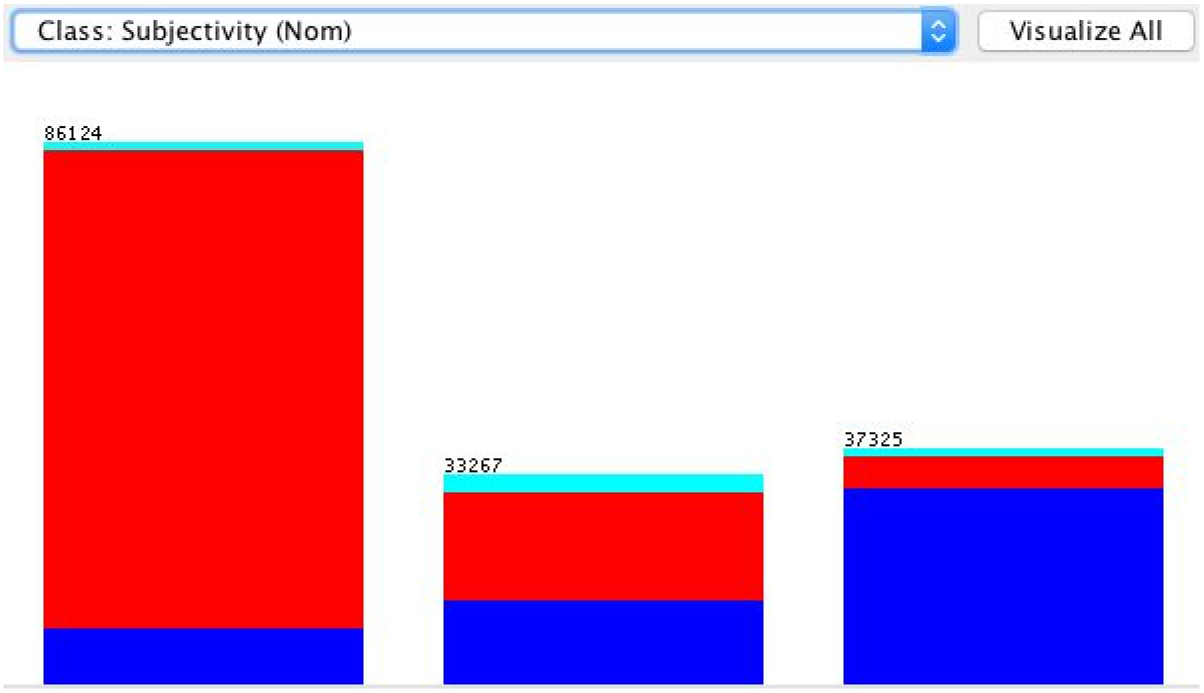
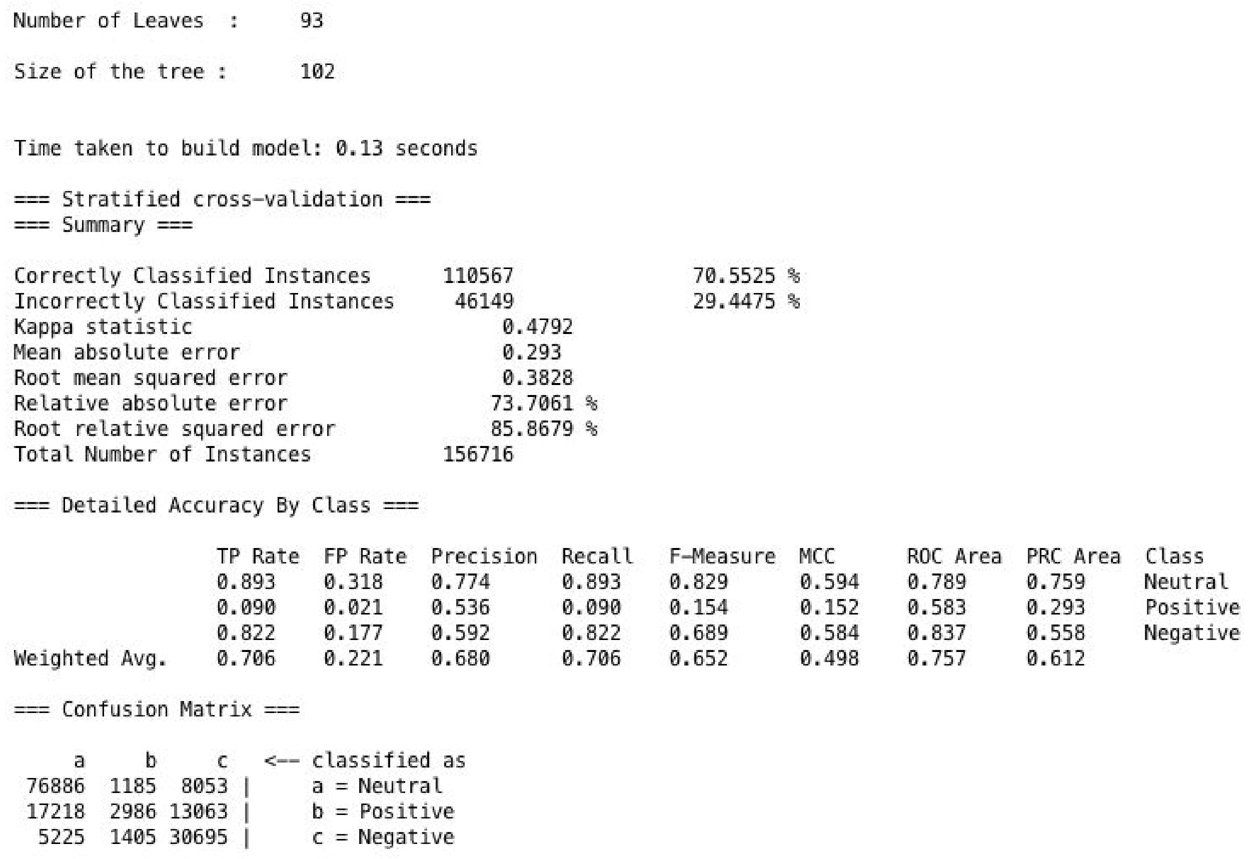


Fig.5.4.4 Number of different subjectivity tweets at different polarities

We then tried to apply machine learning algorithms to the data set. For here, we choosed the polarity of the tweets as the target label. A standard

J48 decision tree model was used in this case. We applied a 10 folds cross-validation to observe the result (fig.5.4.5). The result showed that the accuracy is 70.5525% which is not very high. We applied Bayes Net model to do the same prediction and got 70.2634% accuracy. We also tried to do the disease trend prediction at this time, but due to the lack of feature, no useful information was found during the exploration.



```
Number of Leaves  :     93

Size of the tree :      102


Time taken to build model: 0.13 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances      110567               70.5525 %
Incorrectly Classified Instances     46149               29.4475 %
Kappa statistic                          0.4792
Mean absolute error                      0.293
Root mean squared error                  0.3828
Relative absolute error                 73.7061 %
Root relative squared error             85.8679 %
Total Number of Instances           156716

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 0.893    0.318    0.774      0.893    0.829      0.594    0.789     0.759     Neutral
                 0.090    0.021    0.536      0.090    0.154      0.152    0.583     0.293     Positive
                 0.822    0.177    0.592      0.822    0.689      0.584    0.837     0.558     Negative
Weighted Avg.    0.706    0.221    0.680      0.706    0.652      0.498    0.757     0.612

=== Confusion Matrix ===

     a     b     c   <-- classified as
 76886  1185  8053 |     a = Neutral
 17218  2986 13063 |     b = Positive
  5225  1405 30695 |     c = Negative
```

Fig.5.4.5 Experiment results using WEKA

6. Conclusion and Future Work

We collect flu data using Twitter APIs, do the date cleaning and spam elimination, and then do the basic analysis based on time periods and geolocations, also implement the sentiment analysis and machine learning exploration to do flu sentiment prediction.

We can derive some conclusions from the experiments results. From the analysis based on time period, we can derive that the flu symptoms and spreading are worse in the noon, because there are more tweets about flu were posted at noon; Also we find out that the number of flu tweets and population has a positive correlation, most tweets are from California, this is basically because CA has the largest population; from the sentiment analysis, we could propose that the number of positive and negative sentences has no significant difference over all. But seeing into the subjective feature, we found that the subjective tweets are basically all made up of negative ones. Of course the NLP api could have a significant influence on this. We are able to test other sentiment apis to have a conclusion on this. From the machine learning we have learned that the sentiment prediction is feasible.

We plan in the next few months to do more analysis; for example comparing the emergence and spread of flu and other common infectious diseases such as mumps or Dengue fever, or tracking and predicting the emergence and spread of seasonal influenza epidemics in tweets in different languages. Also, we are looking forward to grab more features from tweets and applied some more advanced machine learning algorithms to predict the trend of such diseases.

7. References

[1] F. Jordans, “WHO working on formulas to model swine flu spread,” 2009.

[2] N. M. Ferguson, D. A. Cummings, S. Cauchemez, C. Fraser, S. Riley, A. Meeyai, S. Iamsirithaworn, and D. S. Burke, “Strategies for containing an emerging influenza pandemic in southeast asia,” Nature, vol. 437, 2005.

[3] I. Longini, A. Nizam, S. Xu, K. Ungchusak, W. Hanshaoworakul, D. Cummings, and M. Halloran, “Containing pandemic influenza at the source,” Science, vol. 309, no. 5737, 2005.

[4] A. Sitaram and B. A. Huberman, “Predicting the future with social media,” in Social Computing HP Lab, Palo Alto, USA, 2010.

[5] T. Sakaki, M. Okazaki, and Y. Matsuo, “Earthquake shakes twitter users: real-time event detection by social sensors,” in 19th international conference on World wide web, Raleigh, North Carolina, USA, 2010.